Programmation avancée

Répétition 7: Résolution de problèmes

Jean-Michel Begon

30 novembre 2018

1 100-chaîne

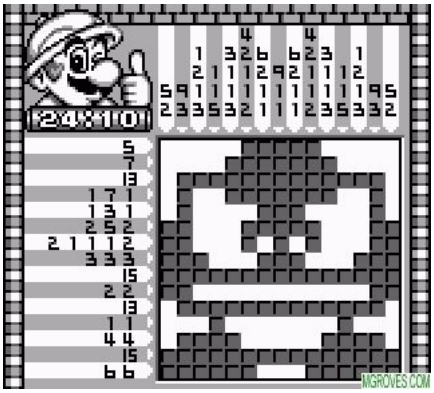
Soit la chaîne "123456789". Proposez un algorithme qui énumère les différentes manières d'insérer des "+" et des "-" entre les *nombres* de manière à obtenir un total de 100.

Par exemple, $123 + 45 - 67 + 8 - 9 = 100$.

Quelle est la complexité de votre solution ?

2 Logimage (aussi appelé picross)

Proposez un algorithme par recherche exhaustive pour résoudre un logimage :



Cette solution est-elle envisageable en pratique ? Par exemple pour une grille 15×15 ?

3 Les 12 pièces

Soit le problème suivant :

> *"Vous disposez de 12 pièces dont une est fausse. Celle-ci possède un poids moindre. Comment déterminer la fausse pièce le plus rapidement possible à l'aide d'une balance à plateaux ?"*

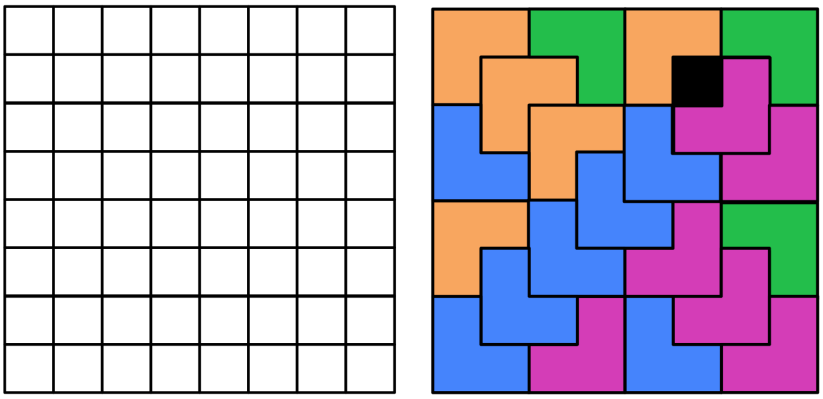
Dans le cadre général où on dispose de N pièces :

1. Proposez une approche par force brute pour résoudre ce problème.
2. Proposez une approche *divide-and-conquer* pour résoudre ce problème. Peut-on donner une borne sur le nombre de pesées ?

4 Carré de triminos

On souhaite recouvrir un carré de taille NxN ($N = 2^m, m \geq 0$) à l'aide des quatre types triminos, à l'exception d'une case vide dont les coordonnées sont données.

Les triminos de base



Carré 8x8 vide et recouvert

Proposez un algorithme pour résoudre ce problème.

5 Distance minimum

Soit un ensemble $P = \{p_1, p_2, ..., p_n\}$ de points ($n = 2^k$, $k > 0$) répartis dans un plan et tels que $p_i = (x_i, y_i)$. On cherche un algorithme qui détermine la distance (euclidienne) entre les deux points les plus proches.

1. Proposez une approche exhaustive pour solutionner ce problème.
2. Proposez un algorithme diviser-pour-régner pour ce problème.
3. Comparez la complexité des deux algorithmes.

6 Multiplication matricielle

Proposez un algorithme diviser-pour-régner afin de multiplier deux matrices $n \times n$ ($n = 2^k, k \geq 0$). Quelle est la complexité de la solution ?

7 Nombres de Catalan

Dans une répétition précédente, nous avions solutionné l'exercice :

> *Soit un ensemble de N valeurs entières distinctes. Ecrivez une fonction calculant le nombre d'arbres binaires de recherche distincts qu'il est possible de construire à partir de ces N valeurs ($N \geq 1$).*

par le pseudo-code suivant :

```
NbTree(N)
1  if N ≤ 1
2        return 1
3  Nb = 0
4  for i = 1 to N
5        Nb = Nb + NbTree(i − 1) ∗ NbTree(N − i)
6  return Nb
```

dont la complexité est importante à cause des appels à des sous-problèmes déjà résolus.

1. Dessinez le graphe des sous-problèmes pour une taille de $N = 4$.
2. Utilisez la mémoïsation pour faire baisser la complexité de l'algorithme (donnez le pseudo-code d'une version ascendante et d'une version descendante).
3. Quelles sont ces complexités ?